GENERALIZING SINGLE-VIEW 3D RECONSTRUCTION

A Dissertation
Presented to
The Academic Faculty

By

Randal Michnovicz

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Science in the
School of School of Computer Science

Georgia Institute of Technology

May 2020

GENERALIZING SINGLE-VIEW 3D RECONSTRUCTION

Approved by:

Dr. James Rehg, Advisor
School of Interactive Computing
Georgia Institute of Technology

Dr. Zsolt Kira
School of Interactive Computing
Georgia Institute of Technology

Date Approved: April 30, 2020

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

3D reconstruction, a task where algorithms generate 3D models from images taken from an ordinary camera, is a classic problem in computer vision. Approaches to generate 3D models from many images, such as bundle adjustment and structure from motion, were invented before deep learning [1] [2]. However, some more recent work focuses on the ability to construct 3D models from a single image — a difficult task, especially as algorithms do not get to see the back side of the object [3] [4] [5]. With no explicit depth or geometry data, these approaches must use priors to make inferences about 3D shape. Algorithms to generate 3D meshes have shown impressive performance on this task. However, Tatarchenko et al. [6] found that these algorithms essentially memorize the shapes in the training data, and only perform well on shapes in testing data due to the similarity of the two sets, and cannot generalize to dissimilar shapes. Shin et al. [7] found that this memorization happens in part because models are expected to reconstruct an object in a certain pose provided by a dataset, specifically in an object-centered coordinate system, rather than reconstructing the object in the same pose that was used to render the image, or in a viewer-centered coordinate system. This issue is one of many problems and inconsistencies that plague the field, which also include inconsistencies in object pose in renders, pointcloud generation, and train/test splits. In this work, we experimentally investigate the effects of train/test splits, architectures, and shape dataset choices as part of proposing a unified approach for the field of single-image 3D reconstruction. We combine several components of previous approaches to create an architecture which establishes a state-of-the-art baseline for our approach. The contributions are as follows.

- We create a method to split a dataset to minimize leakage and then compare its performance metrics to a dataset split used in previous work. I wrote the code to split the

datasets while consulting Stefan Stojanov, and Anh Ngoc Thai ran the experiment.

- We train and test our architecture and compare it to other architectures, and demonstrate state-of-the-art performance. This work was performed in conjunction with Anh Ngoc Thai, Stefan Stojanov, and Vijay Upadhya. I contributed to this by providing argument about how objects should be centered in images.

- We train and test our architecture within and across the datasets Shapenet [8] and ABC [9] and report results for both visible and invisible surfaces. This work was performed in conjunction with Anh Ngoc Thai, Stefan Stojanov, and Vijay Upadhya. I contributed to this by helping Stephan Stojanov devise a method to texture ABC shapes.

CHAPTER 2

BACKGROUND

2.1 Shape Datasets and Normalization

There are many shape datasets available. We use ShapeNetCore [8] due to its popularity and size. Shapenet has over 50,000 meshes and associated materials spread across 55 categories, the majority of which involve types of manmade objects one would find inside a home. Other datasets are less structured, such as the 1,000,000+ model dataset ABC [9], which seems to have a lot of models used for manufacturing. Since reconstruction algorithms do not seek to find the overall size of the object, shapes in these datasets are standardized in size by scaling the longest dimension of the shape to be of 1 unit length, and the center of mass of the object at the axis. In addition, since meshes may not be watertight, a prerequisite for volume-based analysis, we use tools such as Blender to fill mesh holes. These datasets do not contain train/test splits; we propose a standardized method to do so in our methodology.

2.2 Representations and Metrics

There are three main ways that existing works represent shape: a mesh of 2-D shapes, most often triangles, a pointcloud of points in 3-D space uniformly sampled on the surface of a mesh, and a volumetric representation such as a voxel grid or a voxel grid simplified in an octree. While triangle meshes are the main representation for 3D shapes, comparing them directly is not straightforward. Thus the following are metrics that use pointclouds and voxels.

- Voxel-based (or otherwise binarized volume based) intersection over Union (IoU), also known as Jaccard Index, is a common metric for volumetric representations, as

it is intuitive and focuses on the "filled" space of each object. Its values range from 0 for completely dissimilar objects to 1 identical objects. It is also notable that 1 - IoU, or Jaccard Distance, is a valid distance metric [10]. While IoU captures a difference in global shape, voxel-based representations do not capture the difference in surface details very well, as they focus on the overall volume of the objects.

- Chamfer distance (CD) is a metric for comparing pointclouds. For two pointclouds, the CD is the average of the euclidean distances from each point in each pointcloud to its nearest neighbor in the other pointcloud. Two identical pointclouds will have a CD of 0, and the maximum possible CD is $\sqrt{3}$.

- Normals Consistancy (NC) is another pointcloud-based metric. For two pointclouds with associated normal vectors, the NC is the average of the following products for each point in each pointcloud: the point's normal vector times the normal vector of the point's nearest neighbor on the other pointcloud. Thus, identical pointclouds will have a NC of 1, and NC cannot be less than 0. NC is good for comparing the surface details between two meshes; it has little meaning when two shapes are not globally similar.

- F-score of pointclouds is the metric proposed by Tatarchenko et al. [6]. It defines precision as the fraction of points in a reconstruction's pointcloud that lie within a certain distance to any point in the ground truth's pointcloud; recall is the fraction of points in a ground truth's pointcloud that lie within a certain distance to any point in a reconstruction's pointcloud. When discussing the F-score of two shapes, we must define this distance threshold d. We define FS@1% to the the f-score of two objects with the distance threshold defined to be .01 units.

One inconsistency in existing work using pointcloud-based metrics is the number of samples used in the pointcloud. This is important because in both of the previous pointcloud-based metrics, if there are fewer points in the pointclouds of two objects, the two objects

will appear more dissimilar. Thus, we investigate how the size of two pointclouds generated from identical objects can affect the above distance metrics. In addition, we show how simplifying object meshes can affect the metrics to gain a more intuitive explanation of how these metrics work. In particular, our work is some of the first investigating how the distance threshold of F-score affects the ability for the metric to measure surface detail.

2.3 Object Views

Shin et al. [7] investigates the difference between posing the challenge of shape reconstruction in both the object-centered (OC) and viewer-centered (VC) coordinate frames. In most prior work, 3D reconstruction is posed as a object-centered challenge: given a picture of a 3D object taken at an angle from a random azimuth and elevation, reconstruct the original object. For shapenet, this requires the algorithm to know the "front" of the object, which points towards the X axis, and the "top" of the object with points towards the Z axis. Shin et al. [7] and Tatarchenko et al. [6] provide evidence that object-centered 3D reconstruction algorithms essentially memorize input data in part so that they can guess the correct orientation of objects. Instead, Shin et al. [7] shows that using viewer-centered coordinates provide better reconstructions for unseen shapes and categories, while object-centered coordinates provide better reconstructions for previously seen shapes. Our work builds on existing experiments by posing reconstruction in a slightly different way. Existing approaches only vary azimuth and elevation of the camera, resulting in the vertical axis of an object's canonical pose to be aligned to the y axis of an image. By varying the *tilt* of the camera, rotating the camera about the line perpendicular to its virtual lens, we obtain a more diverse set of views on the object. Thus, images may appear tilted relative to gravity. We dub the approach to the task with this third degree of camera freedom random-VC, which we differentiate from the existing approach by calling that canonical-VC. We standardize the way objects are centered using the center of an axis-aligned bounding box.

2.4 Existing Models

Many approaches to single-view 3D reconstruction exist. Some simply attempt to reconstruct the voxels of a shape, some generate pointclouds to be made into meshes, and several deform a 3D mesh into a predicted 3D mesh. We highlight the models that we use in our experiments. These models inspired our own.

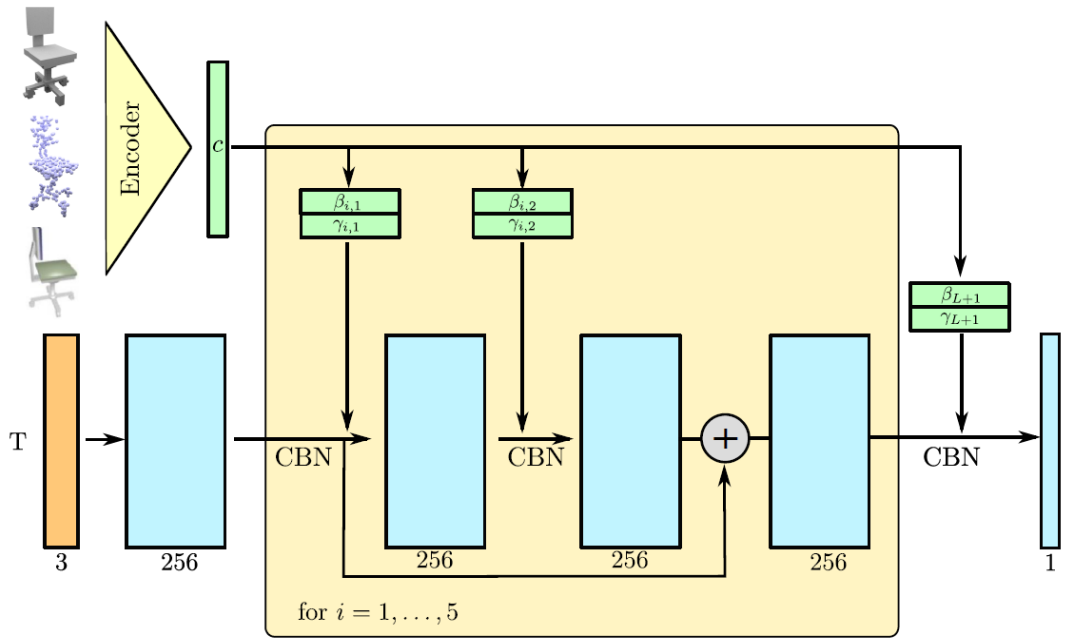

Figure 2.1: OccNet [4] as presented in its source paper. Images are excracted as γ and β intermediate features for conditional batch normalization (CBN). A ResNet-based network predicts binary occupancy by incorporating image-based features in CBN layers.

- OccNet [4] uses an encoder-decoder architecture to predict whether randomly sampled 3D points are contained inside or outside a mesh, that is, it predicts the points' binary *occupancies*. In the first part of the model, an image is fed into an imagenet-pretrained ResNet-18 network ending with a fully connected layer to output a 256-dimensional vector C. A 2-layer fully connected layer generates the γ and β parameters for conditional batch normalization in the second part. The second part of the network classifies occupancy of each point, outputting values in the range [0, 1] to

represent occupancy for each point. This part of the network is made up with a 5-block resnet which uses a conditional batch normalization and ReLU layer between each 256-dimensional fully connected layer. The last layer is fed to a 1-dimensional layer which is used to generate each point's occupancy. The occupancy output is binarized using a threshold value of $\tau = 0.2$.

These occupancies are used to construct a mesh using a method the paper dubs Multiresolution IsoSurface Extraction (MISE). It uses the occupancies to make a 128^3-resolution voxelized surface. The voxels are converted to a triangle mesh using the marching cubes algorithm, simplified using the Fast-Quadric-Mesh-Simplification algorithm, and then refined using the second-order gradient information.

- DISN [3], like OccNet, uses an encoder-decoder architecture that can process arbitrary points. However, unlike OccNet, instead of predicting whether or not a point is contained in a mesh, it predicts a point's Signed Distance Field (SDF), which is a point's distance to the surface of the object, signed positive if the point is outside of the object or negative if the point is inside of the object. The SDF of a point can contain more information about the surrounding than a binary occupancy. It also differs from OccNet by containing two decoders to estimate a point's SDF: one decoder that simply uses the global shape of the object, and one uses an estimated camera angle to guess which features are important to find the SDF for a point.

- GenRe [5] is a model in a paper that tests generalization performance on viewer-centered representations of 10 shapenet categories, trained on only 3 other shapenet categories. It uses a multi stage architecture, which can be described in the following stages with neural nets in between each stage: image $\rightarrow$ depth map $\rightarrow$ spherical map $\rightarrow$ voxels $\rightarrow$ voxel refinement network. Our implementation differs from the paper's because we align a mesh to the ground truth mesh before voxelizing it rather than after voxelizing it. This change removes the artifacts created from rotating voxel

grids.

- Multi-View [7] has an encoder-decoder architecture that takes an input of a depth map and silhouette rather than an RGB image. It contains 10 different encoders that all encode a depth map from a different object view, a depth map of the opposite object view, and a silhouette. The depth maps must be combined to create a mesh.

CHAPTER 3
METHODOLOGY

We conduct several experiments to investigate how different views effect performance and generalization. The first experiment investigates the effect of F-score threshold and point-cloud density on sampling floor. The second experiment compares how a train/test split from GenRe compares in test metrics to a train/test split designed to miminize leakage. The third experiment finds how different views affect generalization. The fourth experiment investigates how our model performs compared to other datasets for the canonical VC task. Finally, the last experiment compares results of cross-dataset generalization to within-dataset generalization for both visible and invisible object surfaces. We report CD, NC, IoU, and FS@1% for every experiment. The datasets, the model, and the experiments are described in further detail below. All models and experiments are implemented using Python and PyTorch, and the code used will be available online.

3.1 Dataset

In our experiments, we use the entirety of ShapeNet [8], which includes over 50,000 models across 55 different categories. All pointcloud-based metrics are computed using 300,000 points randomly sampled on the model's surface, except where otherwise noted. IoU metrics are calculated using 64^3 resolution voxelizations. All reconstruction experiments are trained using a dataset consisting of 25 different 300x300 resolution images, depth maps, and normals maps for each image generated in Blender [11] using the Cycles ray tracing engine.

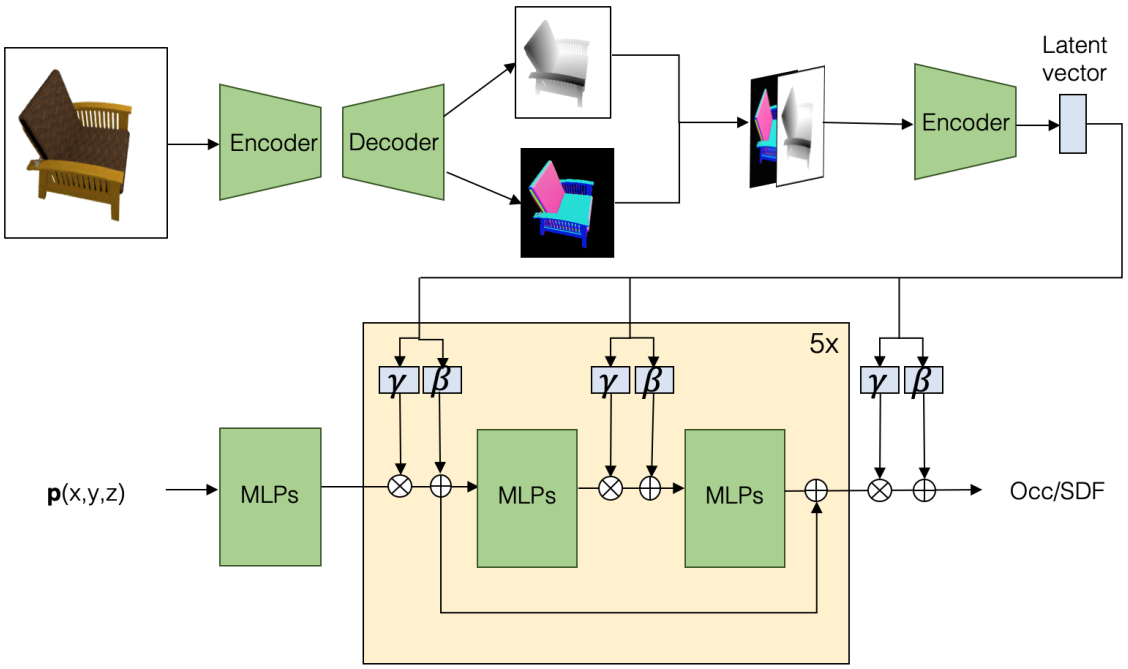

Figure 3.1: Two-stage GenSDFNet architecture. The 2.5D sketch estimator is a ResNet18 architecture with a convolution decoder. Given the depth and surface normal output, the second module produces a feature encoding as in OccNet [4], generating γ and β values for conditional batch normalization between layers. The network is trained to learn Signed Distance Field/Occupancy values for a sampled set of 3D points. Figure Credit: Figure created by Anh Ngoc Thai.

3.2 Architecture

We introduce GenSDF, a model combining aspects of several models that claim state-of-the-art performance on different aspects and approaches to 3D shape reconstruction. The model's architecture contains 2 main components: a 2.5D sketch module that finds the depth and normals maps, and an encoder-decoder architecture which encodes the 2.5D sketch into a latent space, then uses that latent space to predict the SDF (ie. the signed distance from the object's surface) of arbitrary points in 3D space. 3D models are constructed using our model using the MISE algorithm described in Occupancy Networks [4], as described in Related Work. The two components of the model are trained separately using an L_1 loss function and an Adam optimizer with a learning rate of 10^-4. The 2.5D sketch module is taken directly from the codebase of DISN [3], which in turn was taken from

Marrnet [12]. The 2.5D sketch module is an encoder-decoder architecture which encodes images into a latent space using an imagenet-pretrained Resnet-18 module, then decodes the latent space into a 2.5D sketch using 5 5x5 Conv-ReLU layers followed by 4 1x1 Conv-ReLU layers. The second component of the network is identical in its architecture to the encoder-decoder architecture used by OccNet [4], except the first layer of the encoder is modified to take 2 images — the depth map and the normals map — rather than 1 image. This architecture was described in the Related Work section. The difference is that instead of predicting binary occupancy for points in 3D space for a latent representation, we train the model to predict the SDF of points.

3.3 Train/Test Experiment

We split the datasets in 2 different ways and compare the performance on GenSDF on both splits. One way has the 13 most common object categories for a train set and the 42 other object categories for the train set. For the other split, which we refer to as the Oracle Split, we construct the training set using single-linkage heirarchical agglomorative clustering (HAC) initialized with a cluster for each category and IoU as a distance metric. We terminate the algorithm when the largest cluster reaches the size of the other training set. We select the largest cluster to be the training set and the rest of the shapes to be the testing set. This maximizes the minimum jaccard (1-IoU) distance between the testing and training set, thus reducing the potential for dataset leakage. Because our results show the difference between splits is insignificant, we use the 13-42 split in other experiments.

The choice to use single linkage HAC to divide the dataset was driven by the investigation into clustering algorithms suitable for making a train/test split that would be suitable for shape datasets with and without categories. Most clustering algorithms separated a few outlier shapes from the rest effectively, but only HAC the way described provided both a way to control split sizes and pre-cluster shapes into categories.

3.4 Experiment Varying Object Views

We train copies of GenSDF using object-centered, canonical viewer-centered, and randomized viewer-centered views as defined in the Related Work section. We test each copy's performance on both the test set containing canonical views and the test set containing randomized views.

3.5 Experiment Comparing Models

We train and test OccNet, GenRe, and GenSDF on our train-test split for canonical VC.

3.6 Experiment Comparing Visible and Invisible Surfaces

We train and test GenSDFNet on both ShapeNet and ABC, and we test the two resulting models on the two dataset's test sets. We report metrics for objects on both their visible surfaces and their invisible surfaces.

CHAPTER 4

RESULTS

This section discusses how Section 4.1 shows that both of our data splits for Shapenet [8] result in nearly identical performance metrics. Comparing architectures in Section 4.2, we see that our approach achieves state of the art on both seen and unseen classes. Finally, in section 4.3, we compare performance for experiments withing and between the datasets ShapeNet and ABC, and we report performance for both visible and invisible surfaces.

4.1 Train/Test Split Experiment

Table 4.1 shows the difference in model performance when trained on the two splits. Because the two experiments had similar performance, we find that the 13-42 split does not contain significant leakage. We choose to use it in the other experiments.

	Performance on unseen shapes							
	Oracle Split				13-42 Split			
Method	CD	IoU	NC	FS@1	CD	IoU	NC	FS@1
GenSDF	0.38	0.72	0.78	0.44	0.38	0.72	0.83	0.42

Table 4.1: Performance of shape learning methods on ShapeNetCore55. Results do not align to later ones because they were run on an earlier version of GenSDF.

4.2 Experiment Comparing Architectures

In table 4.2, GenSDF demonstrates approximately equal or better generalization performance than existing methods for canonical viewer-centered approaches for metrics. Our method especially performs better in pointcloud distance-based metrics CD and FS@1 .

Both OccNet and SDFNet have far higher IoU values than GenRe, suggesting that approaches using occupancy perform superior on this task in general.

Figure 4.1 shows examples of reconstructed 3D objects from our algorithm. The airplane, which is in the 13 seen classes, looks like a fairly accurate reconstruction. The reconstructions from unseen classes bathtub, guitar, and camera look blobby and miss small details, but they do get the global shape correct.

	Seen				Unseen			
Method	CD	IoU	NC	FS@1	CD	IoU	NC	FS@1
OccNet	0.078	**0.72**	0.78	0.27	0.11	**0.67**	**0.76**	0.22
GenRe	0.181	0.33	0.61	0.10	0.183	0.3	0.61	0.09
SDFNet	**0.05**	**0.72**	**0.79**	**0.41**	**0.08**	0.66	**0.76**	**0.31**

Table 4.2: Performance of different shape learning methods on ShapeNetCore55.

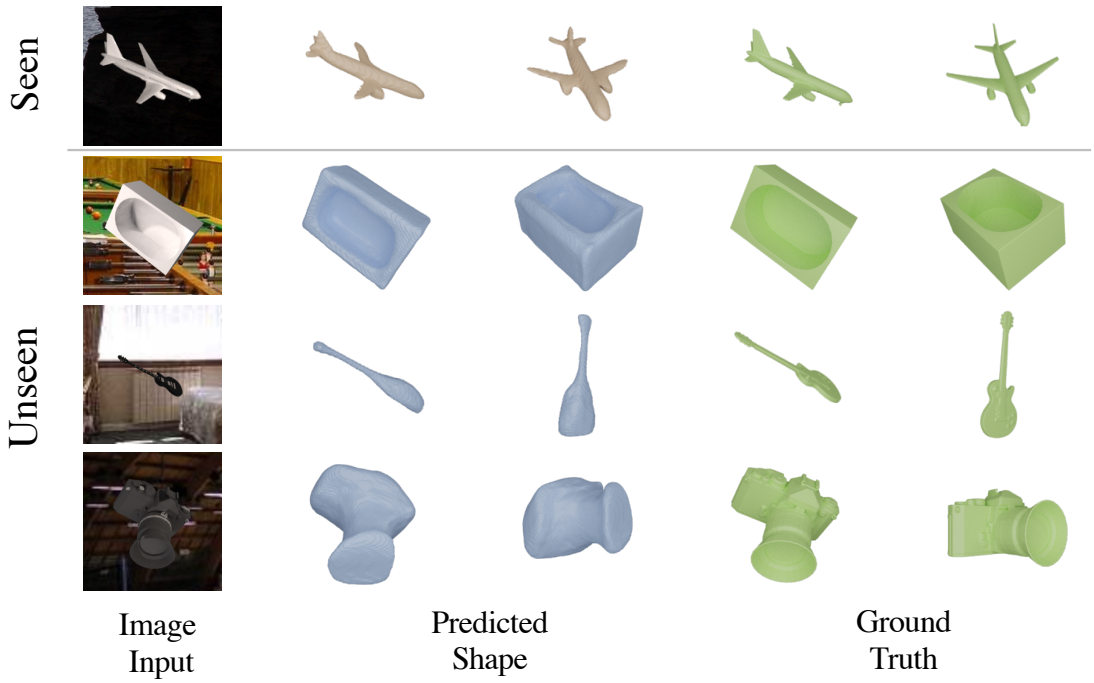

Figure 4.1: Examples of reconstructions of seen an unseen classes from Shapenet using GenSDF. Figure Credit: Figure created by Stefan Stojanov.

4.3 Experiment Comparing Visible and Invisible Surfaces

Table 4.3 demonstrates both the performance within datasets and between datasets for invisible and visible surfaces on objects. The difference between the visible and invisible surfaces in each test is smaller than what may be expected. This suggests that the shape completion part of the architecture may not be taking full advantage of the 2.5D sketch. In addition, the fact that training on ABC and testing on Shapenet suggests that training on this harder, more varied dataset increases generalization ability.

Train data →	ShapeNet				ABC			
Test data ↓	CD	IoU	NC	FS@1	CD	IoU	NC	FS@1
Vis. ShapeNet	0.033	N/A	0.88	0.63	0.038	N/A	0.87	0.57
Invis. ShapeNet	0.058	N/A	0.82	0.38	0.062	N/A	0.81	0.41
Vis. ABC	0.643	N/A	0.73	0.54	0.026	N/A	0.89	0.67
Invis. ABC	0.658	N/A	0.66	0.34	0.44	N/A	0.82	0.55
ShapeNet	0.044	0.75	0.83	0.51	0.047	0.74	0.83	0.50
ABC	0.65	0.64	0.51	0.44	0.035	0.79	0.84	0.62

Table 4.3: Performance of random VC on varied testing and training sets. Results are reported for both the visible and invisible surfaces.

CHAPTER 5
CONCLUSION

Our work introduces a new approach to single-view 3D reconstruction. We standardize our approach to pointcloud sampling in our metrics, object centering, object rotation, and dataset split in hopes that this will bring more unity and direction to the vision community for this task. Our models use of 2.5D sketch estimation and SDF helps it produce state-of-the-art reconstruction performance as measured by every metric discussed.

However, generated models are far from perfect, and our model does not take full advantage of the 2.5D sketch to reconstruct visible surfaces. Future work may include directly using shape symmetry, explicit shape priors, or generative adversarial networks. An approach explicitly using shape symmetry may identify planes in which the shape, or a part of the shape, is predicted to be symmetrical and then constrain the reconstruction based on that knowledge. A modification to our approach to use more explicit shape priors may be able to identify simple shapes in a model like circles and rectangles to guess symmetry, or use the SDF for one part of a predicted 3D model to generate queues for the correct SDF of another part of a 3D model. Finally, the latent space representation in the architecture could be created using a GAN like in Wu et al. [13] so that the algorithm can produce SDF distributions or some other property that are similar to those which appear in the training set.

REFERENCES

[1] X. Han, H. Laga, and M. Bennamoun, "Image-based 3d object reconstruction: State-of-the-art and trends in the deep learning era," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1, 2019.

[2] Y. Chen, Y. Chen, and G. Wang, *Bundle adjustment revisited*, 2019. arXiv: 1912.03858 [cs.CV].

[3] Q. Xu, W. Wang, D. Ceylan, R. Mech, and U. Neumann, "Disn: Deep implicit surface network for high-quality single-view 3d reconstruction," *arXiv preprint arXiv:1905.10711*, 2019.

[4] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3d reconstruction in function space," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.

[5] X. Zhang, Z. Zhang, C. Zhang, J. B. Tenenbaum, W. T. Freeman, and J. Wu, "Learning to Reconstruct Shapes from Unseen Classes," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[6] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, "What do single-view 3d reconstruction networks learn?" In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3405–3414.

[7] D. Shin, C. C. Fowlkes, and D. Hoiem, "Pixels, voxels, and views: A study of shape representations for single view 3d object shape prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3061–3069.

[8] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[9] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "Abc: A big cad model dataset for geometric deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9601–9611.

[10] S. Kosub, *A note on the triangle inequality for the jaccard distance*, 2016. arXiv: 1612.02696 [cs.DM].

[11] Blender Online Community, *Blender - a 3d modelling and rendering package*, Blender Institute, Amsterdam: Blender Foundation.

[12] J. Wu, Y. Wang, T. Xue, X. Sun, B. Freeman, and J. B. Tenenbaum, "Marrnet: 3d shape reconstruction via 2.5d sketches," in *NIPS*, 2017.

[13] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, *Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling*, 2016. arXiv: 1610.07584 [cs.CV].